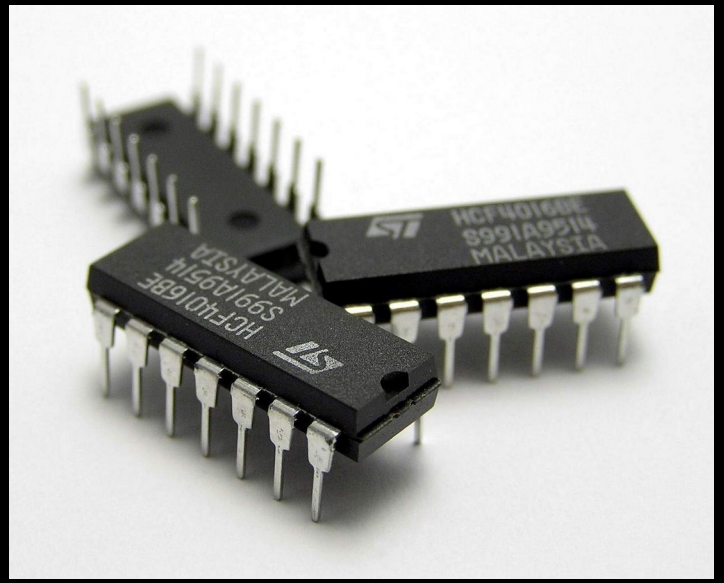
NandSim

Digital Logic Circuit Simulator

Svendeprøve
Simon From Jakobsen
Juni 2026

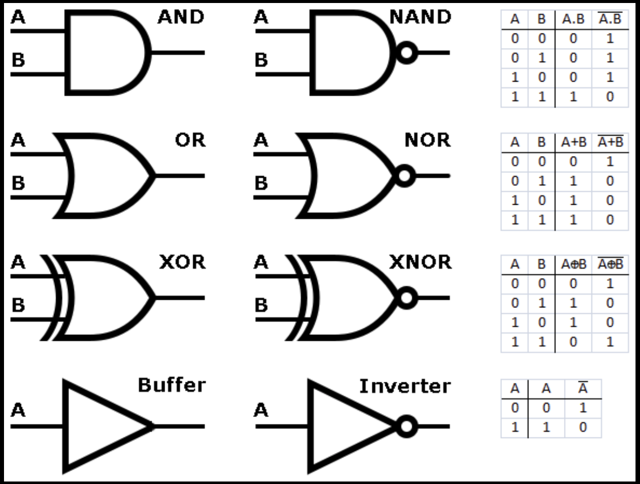
Integrerede kredsløb

- Elektroniske kredsløb
  - ledninger, komponenter, strøm, spænding
- Fotolitografi
- Formål, fx digital-logik
- Samlet pakke
  - inkluderer ofte software



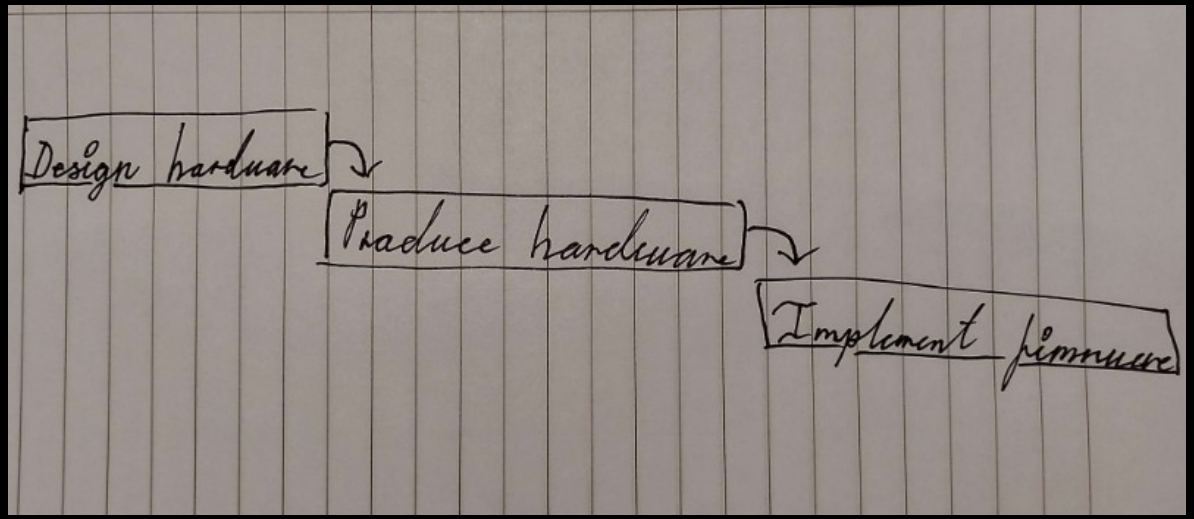
Digital-logik

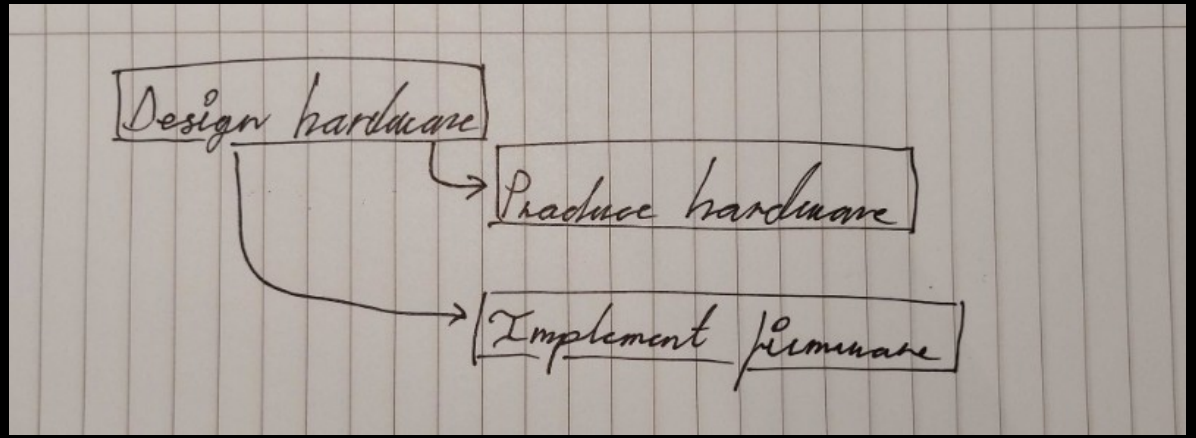
- Digital elektronik
- Binære værdier (1 eller 0, sandt eller falsk…)
- Logik-*gates: AND, OR, NOT, ...*
- Boolesk algebra: ∧ , ∨ , ¬, …
- Komponering
- Computing



Integreret software

- Firmware
- Drivers
- HAL
- RTOS
- Microcode
- Dataset


Design hardware
Produce hardware
Implement firmware

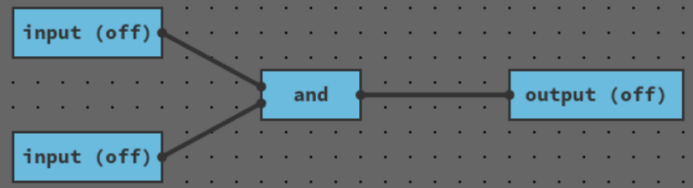

Design hardware
Produce hardware
Implement firmware

Toolbar

- input
- output
- and
- or
- not

(Unnamed) +

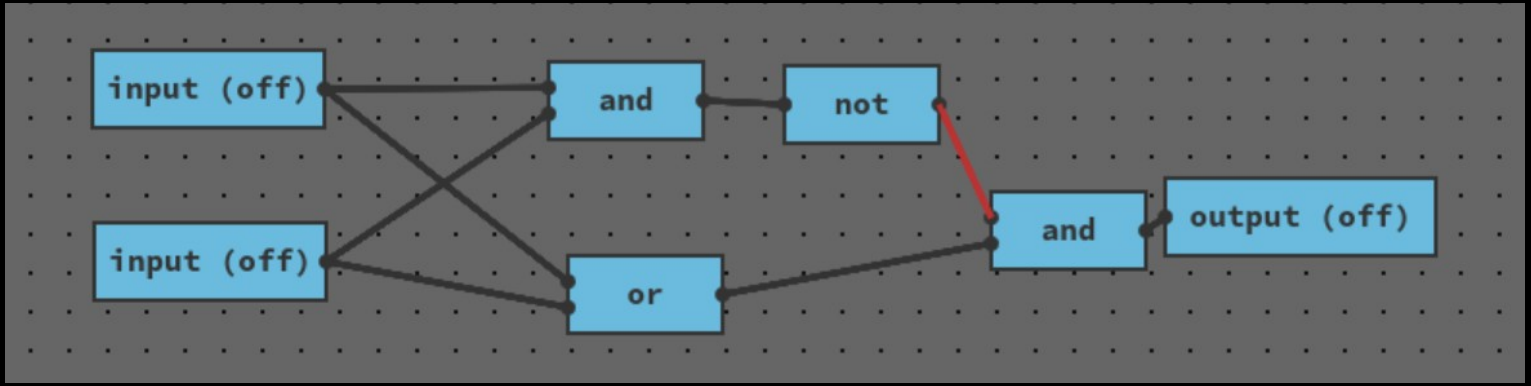
Save Rename Close



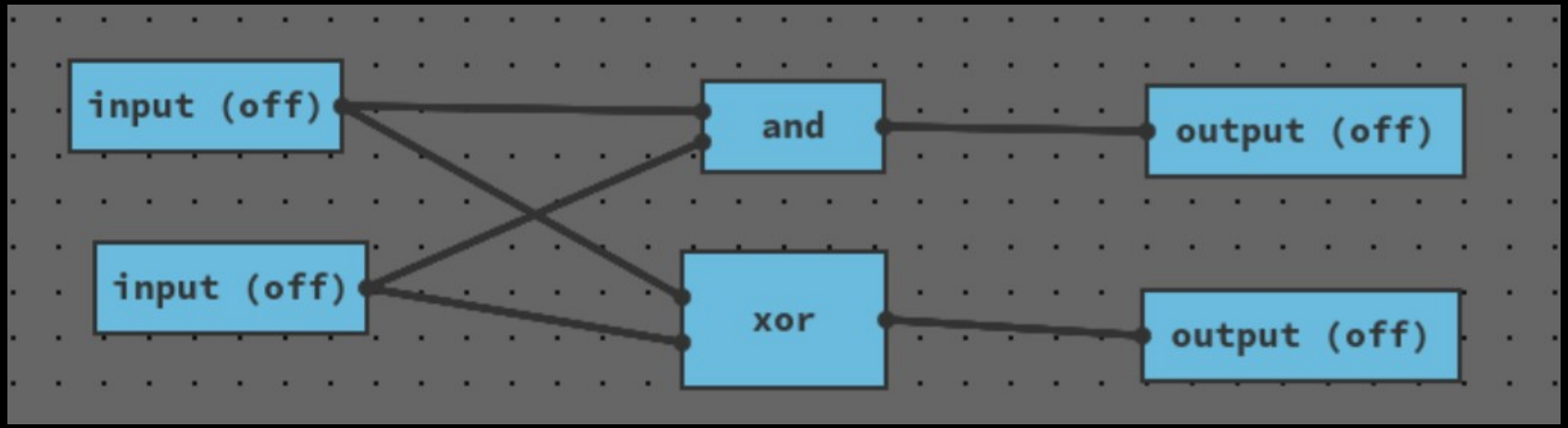
https://nandsim.sfja.dk/

Video af demonstration

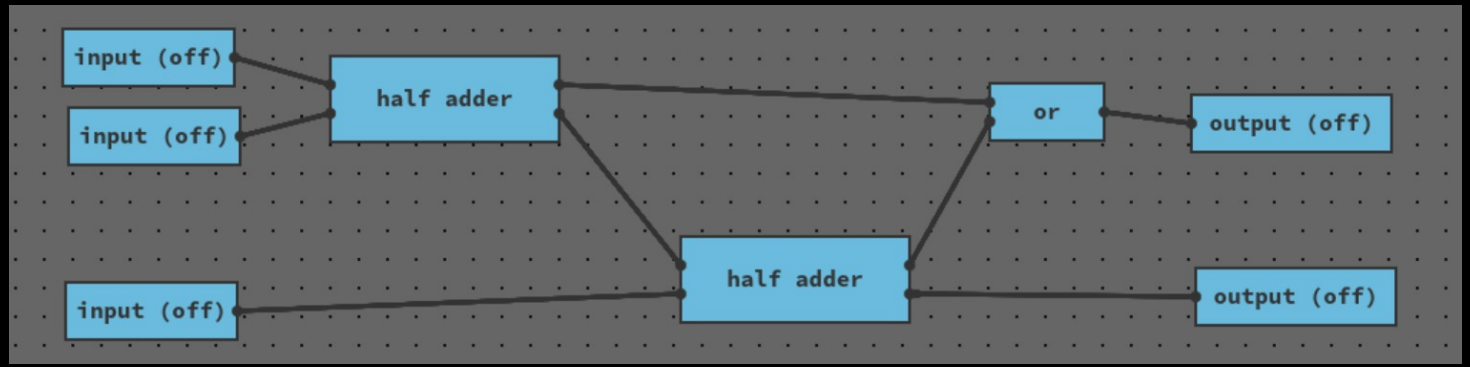
XOR



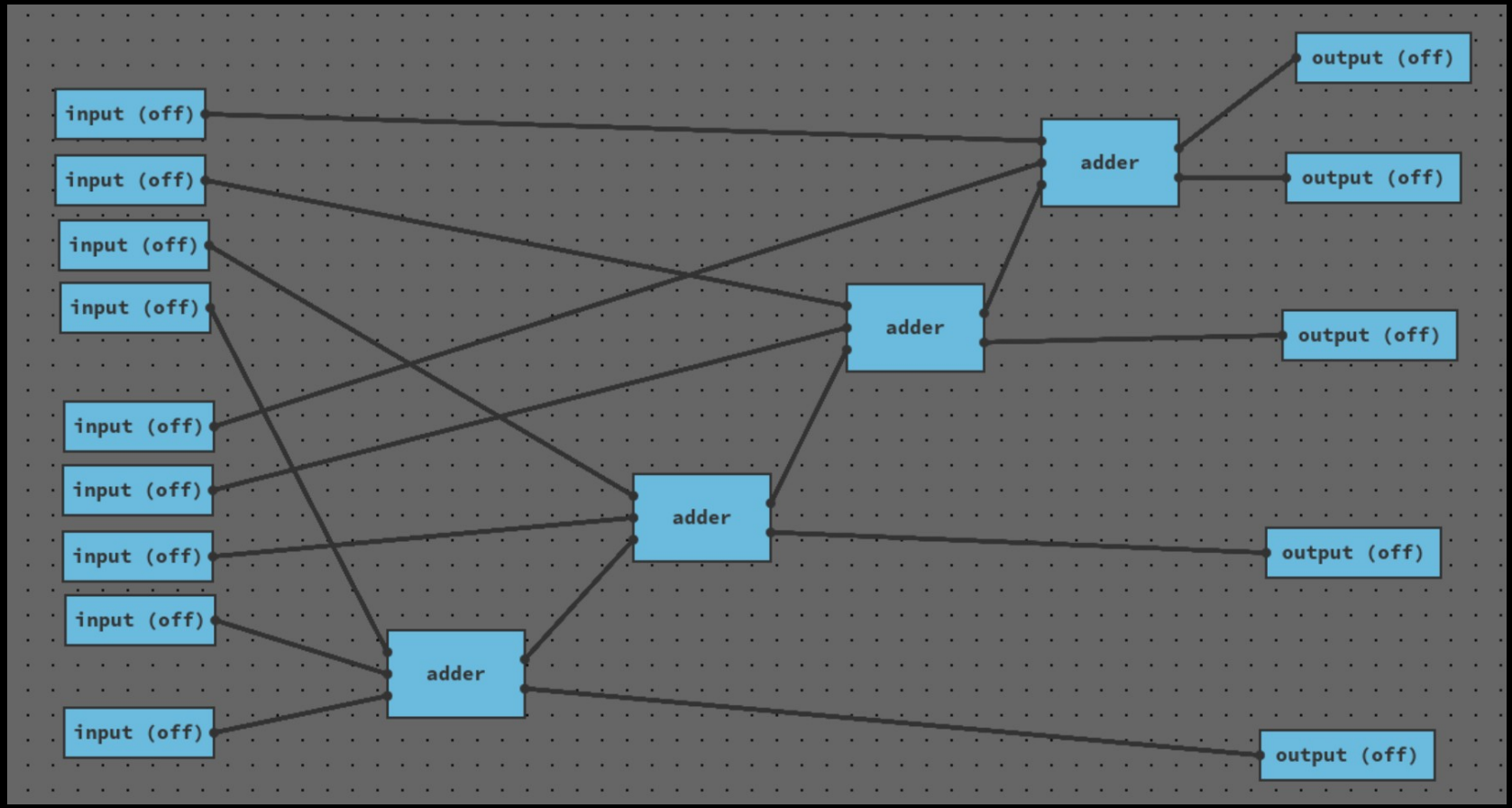
HALF ADDER

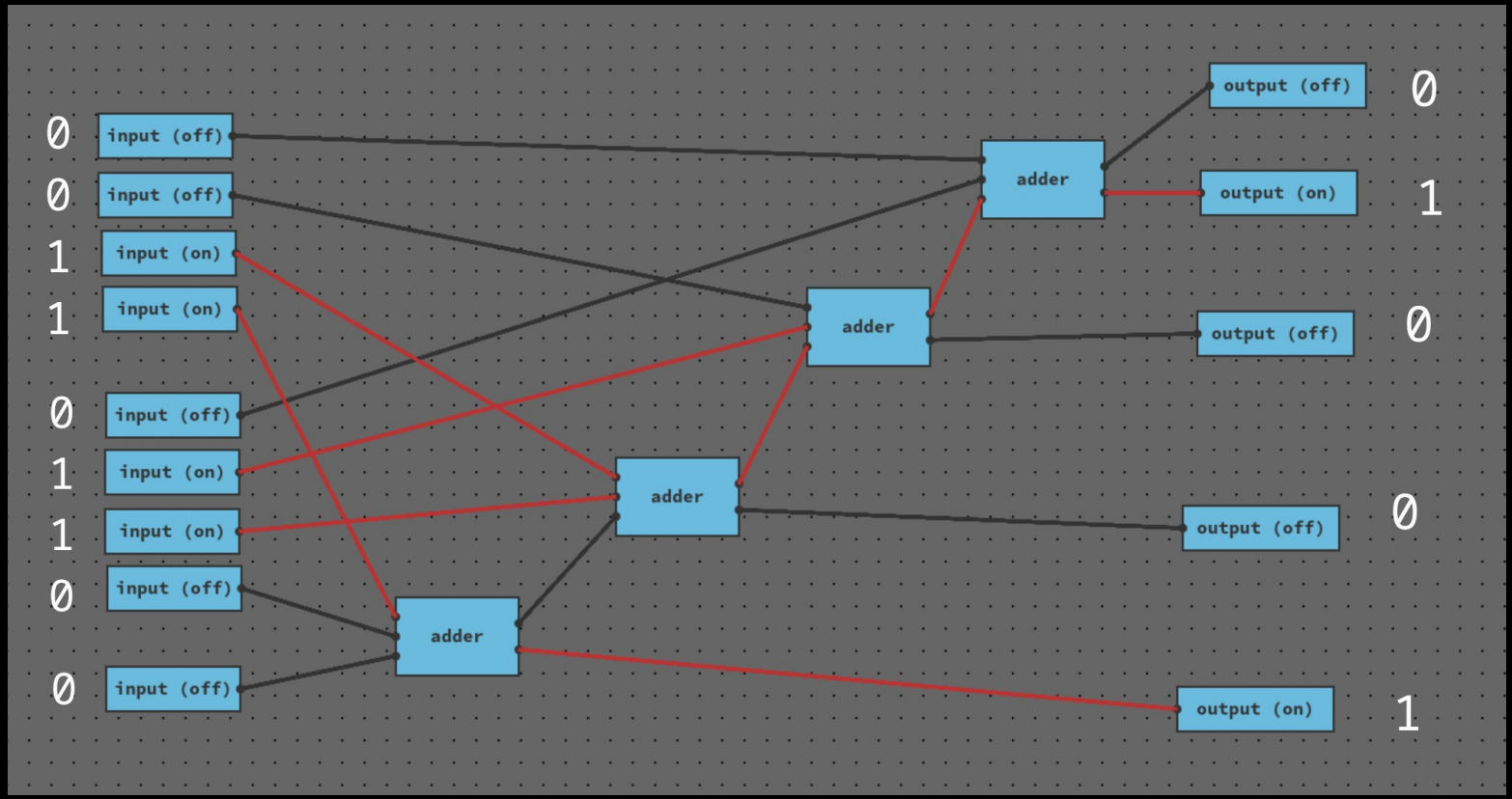


FULL ADDER


input (off)
input (off)
input (off)
half adder
half adder
or
output (off)
output (off)

4-bit ADDER


input (off)
input (off)
input (off)
input (off)
input (off)
input (off)
input (off)
input (off)
input (off)
adder
adder
adder
adder
output (off)
output (off)
output (off)
output (off)
output (off)


0
input (off)
0
input (off)
1
input (on)
1
input (on)
0
input (off)
1
input (on)
1
input (on)
0
input (off)
0
input (off)
adder
adder
adder
adder
output (off)
output (on)
output (off)
output (off)
output (on)
0
1
0
0
1

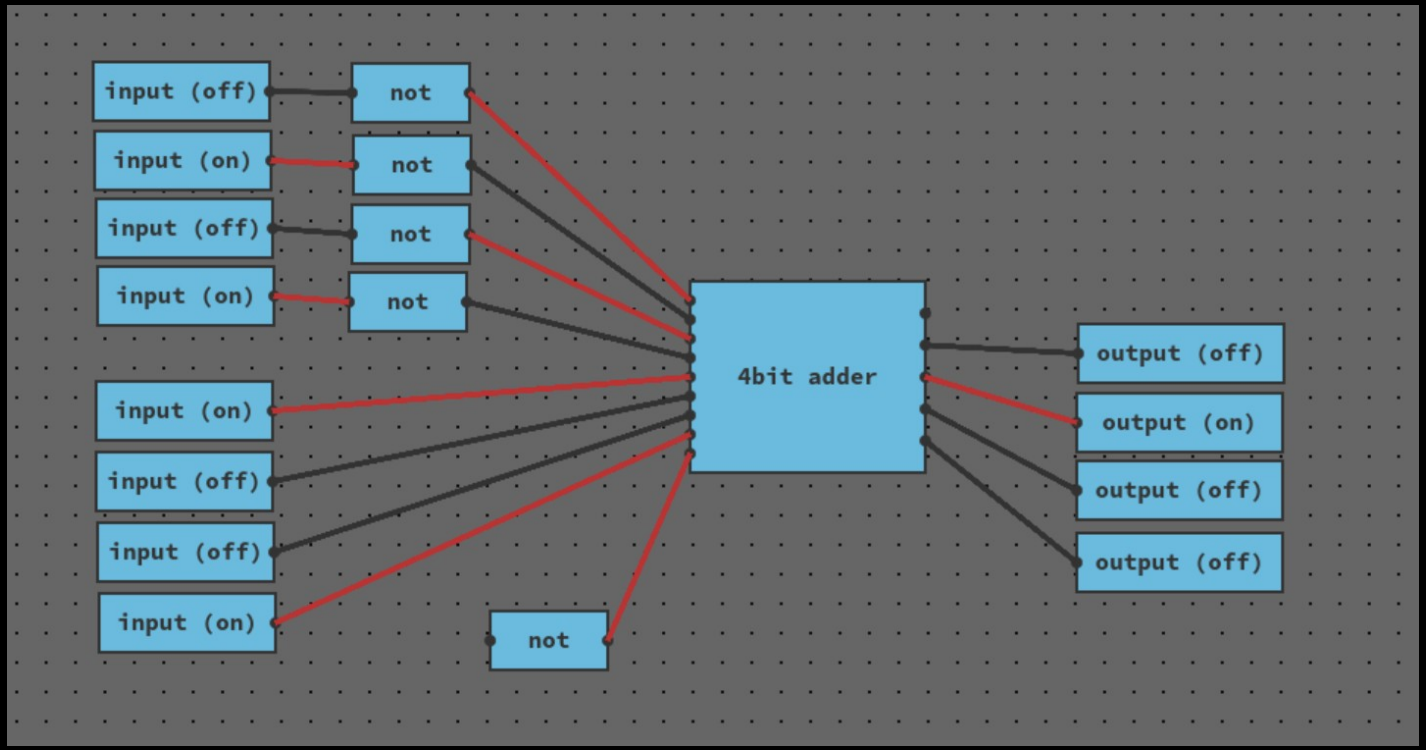
3 = 0011

6 = 0110

```
  0011
+ 0110
------
= 1001
```

1001 = 9

3 + 6 = 9



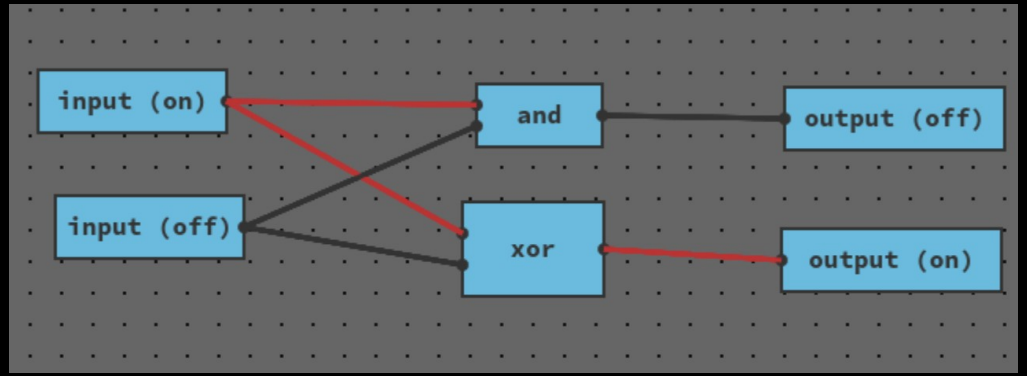
5 = 0101
9 = 1001

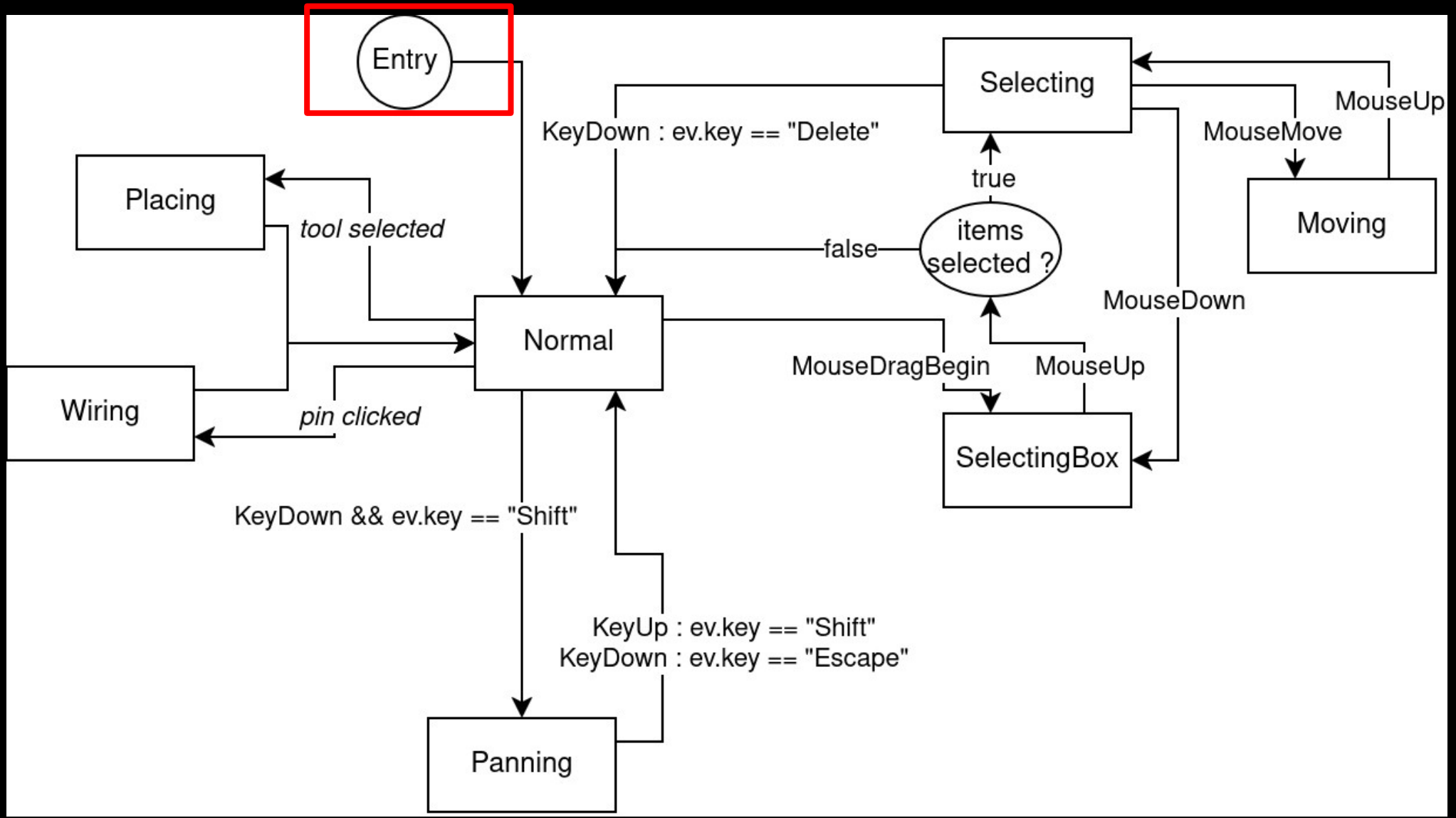
9 - 5 = 4

4 = 0100

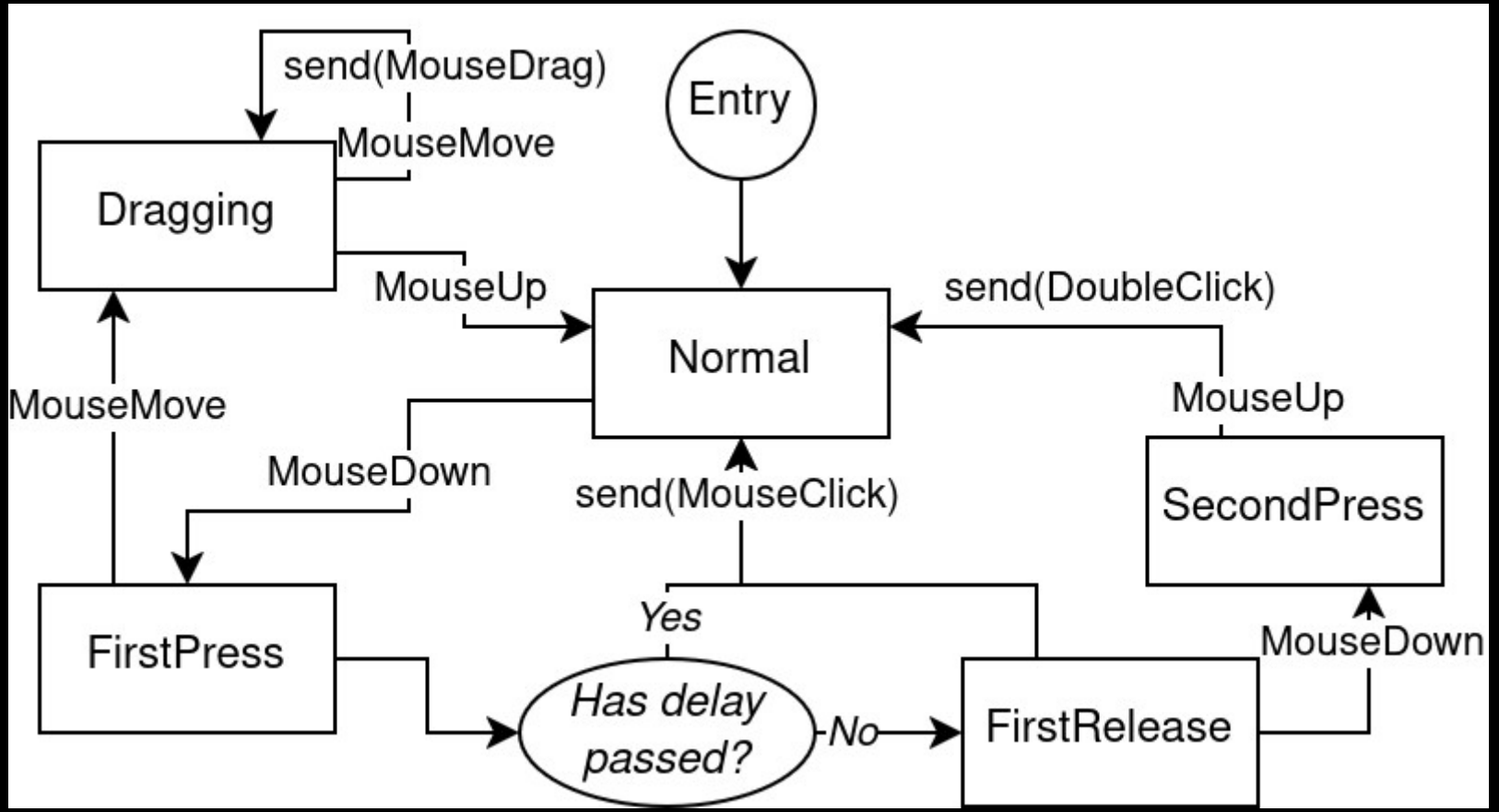
Editor’en

- Visuel
- Kamera
- Selection
- Interaktion
- Flere modes

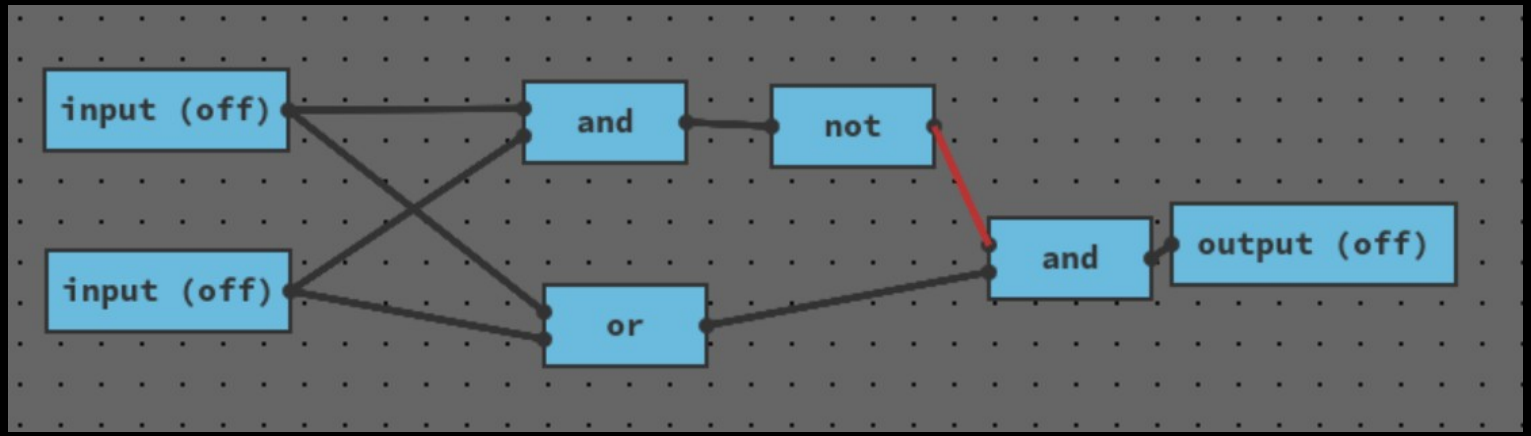



Entry
Selecting
KeyDown : ev.key == "Delete"
MouseMove
MouseUp
Moving
Placing
tool selected
true
items selected ?
false
MouseDown
Normal
MouseDragBegin
MouseUp
Wiring
pin clicked
SelectingBox
KeyDown && ev.key == "Shift"
KeyUp : ev.key == "Shift"
KeyDown : ev.key == "Escape"
Panning

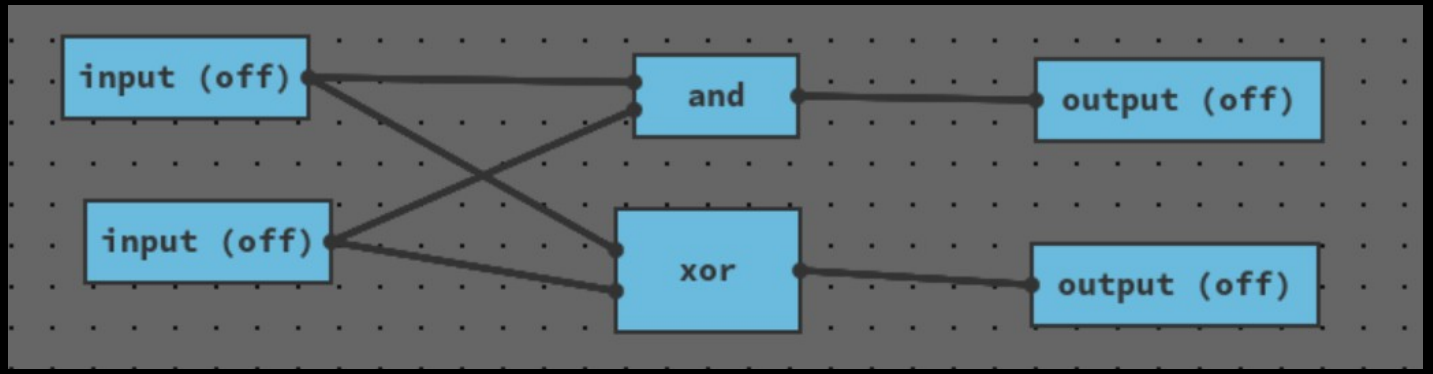
Mouse


send(MouseDrag)
MouseMove
Entry
Dragging
MouseUp
send(DoubleClick)
Normal
MouseMove
MouseUp
MouseDown
send(MouseClick)
SecondPress
FirstPress
Yes
MouseDown
Has delay passed?
No
FirstRelease

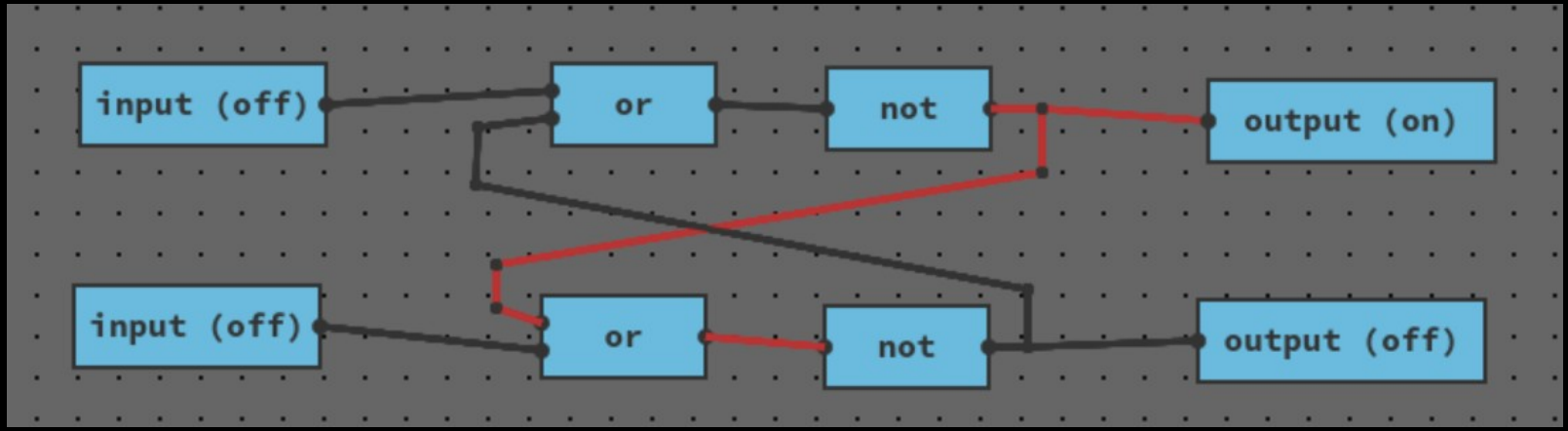
IR / Simulation




```
component xor 2 1 {
    state [  ]
    %0 = Input 0
    %1 = Input 1
    %2 = Nand %0, %1
    %3 = Or %0, %1
    %4 = And %2, %3
    Output 0, %4
}
```




```
component half adder 2 2 {
    state [  ]
    %0 = Input 0
    %1 = Input 1
    %2 = Call xor (%0, %1)
    %3 = Elem %2, 0
    Output 1, %3
    %4 = And %0, %1
    Output 0, %4
}
```



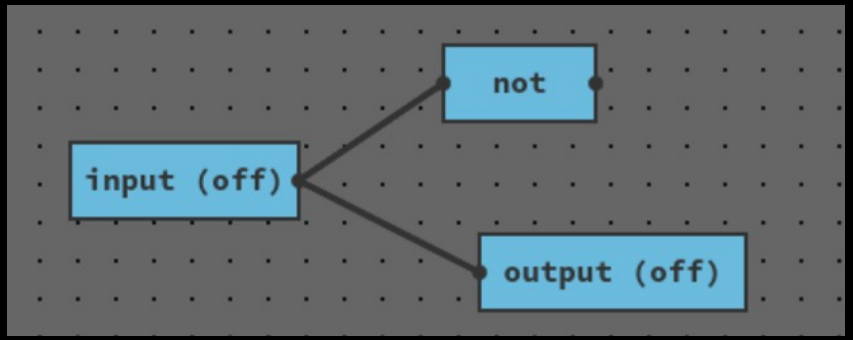

```
component rs latch 2 2 {
    state [ #0 ]
    %0 = Input 0
    %1 = Input 1
    %2 = GetState #0
    %3 = Nor %0, %2
    %4 = Nor %3, %1
    SetState #0, %4
    Output 1, %4
    Output 0, %3
}
```

```
  eliminateUnusedStmts() {
    const mut = new StmtsMutater(this.comp, this.replacedStates);

    const useCount = new Map<Stmt, number>();
    for (const stmt of mut) {
      for (const src of stmt.sources()) {
        useCount.set(src, (useCount.get(src) ?? 0) + 1);
      }
    }

    const nonEffectfulStmts = new Set<StmtKind["tag"]>([
      "And", "Or", "Nand", "Nor", "Elem", "Input", "GetState", "Null",
    ]);

    for (const stmt of mut) {
      if (!useCount.get(stmt) && nonEffectfulStmts.has(stmt.kind.tag)) {
        mut.removeStmt(stmt);
      }
    }
  }
```




```
component <main> 1 1 {
    state [ #0 ]
    %0 = Input 0
    %1 = Not %0
    Output 0, %0
}
```

```
component <main> 1 1 {
    state [ #0 ]
    %0 = Input 0
    Output 0, %0
}
```

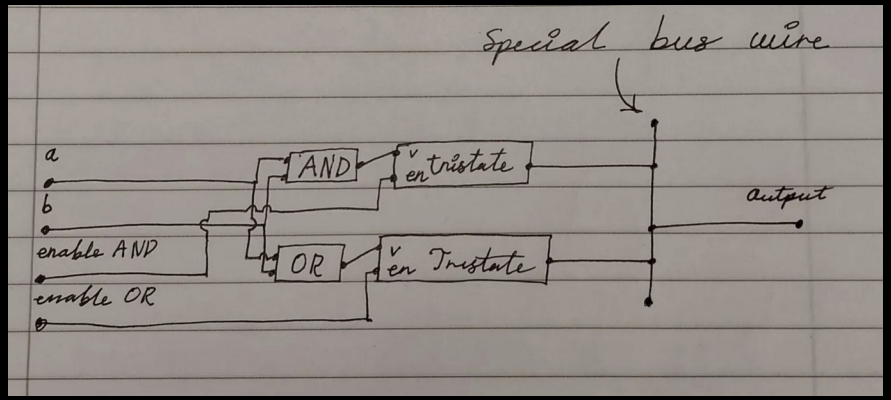
```
  for (let i = 0; i < comp.stmts.length; ++i) {
    const stmt = comp.stmts[i];
    const k = stmt.kind;
    switch (k.tag) {
      case "Null":
        regs[i] = false;
        break;
      case "Input":
        regs[i] = inputs[k.i];
        break;
      case "Output":
        outputs[k.i] = regs[stmtIdcs.get(k.src)!];
        break;
      case "GetState":···
      case "SetState": {···
      }
      case "Not":
        regs[i] = operation((v) => !v, k.op);
        break;
      case "And":
        regs[i] = operation((a, b) => a && b, k.lhs, k.rhs);
        break;
      case "Or":
        regs[i] = operation((a, b) => a || b, k.lhs, k.rhs);
        break;
      case "Nand":
        regs[i] = operation((a, b) => !(a && b), k.lhs, k.rhs);
        break;
      case "Nor":
        regs[i] = operation((a, b) => !(a || b), k.lhs, k.rhs);
        break;
      case "Call": {···
      }
      case "Elem": {···
      }
```

Ting der kan forbedres

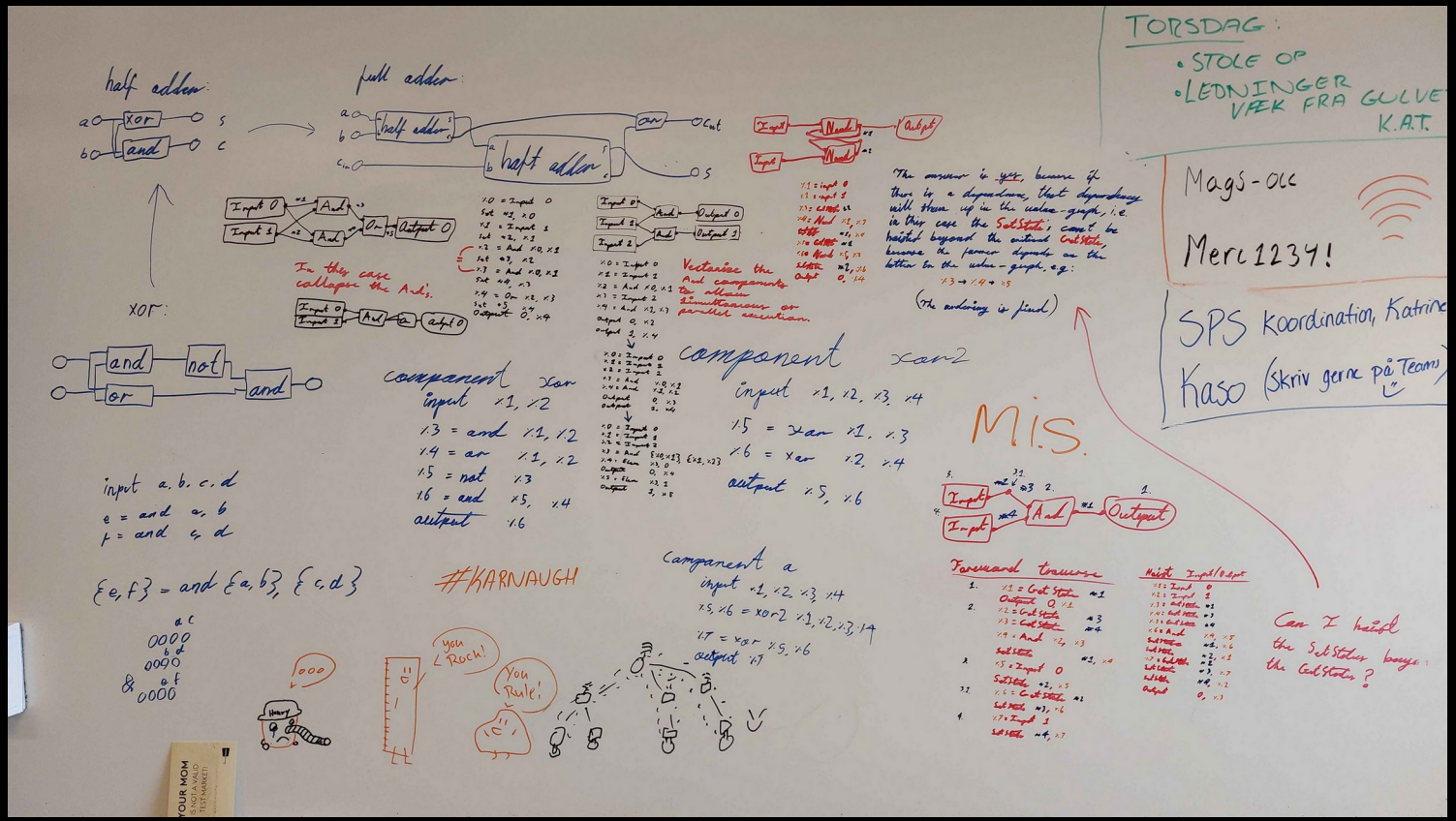
- Musefunktioner virker suboptimalt
- Man kan ikke zoome
- Tabs og save-funktionalitet har huller
- Wire-stilen er grim
- Event-systemet er rodet

Ting jeg ikke nåede

- Importering af data
- Cloud-miljø
- CI-setup
- Native-afvikling
- Low impedance/Tri-state



Konklusion


half adder
full adder
xor
TORSDAG:
• STOLE OP
• LEDNINGER VÆK FRA GULVET K.A.T.
Mags-ole
Merc1234!
SPS koordination, Katrine
Kaso (skriv gerne på Teams)
component xor
component xor2
#KARNAUGH
component a
M.I.S.
Forward traverse
Can I herfra the SubStates bruge the GobStates?